IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Lember Geivi

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies
  - Data collection (API + web scraping)
  - Data wrangling
  - Exploratory data analysis (SQL + graphs)
  - Data visualization (Folium + Plotly Dash)
  - Predictive analysis
- Summary of all results
  - Summary of exploratory data analysis
  - Summary of data visualization
  - Summary of predictive analysis

Introduction

- Project background and context

  Falcon 9 is the world's first partially reusable rocket, which was designed and built by SpaceX. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is coming from the fact that SpaceX can reuse its first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

  The aim of this project is to give an insight to the success of landing the Falcon 9 first stage.

- Problem to find answers

  Try to find, which aspects contribute to the successful first stage landing of Falcon 9.

Section 1

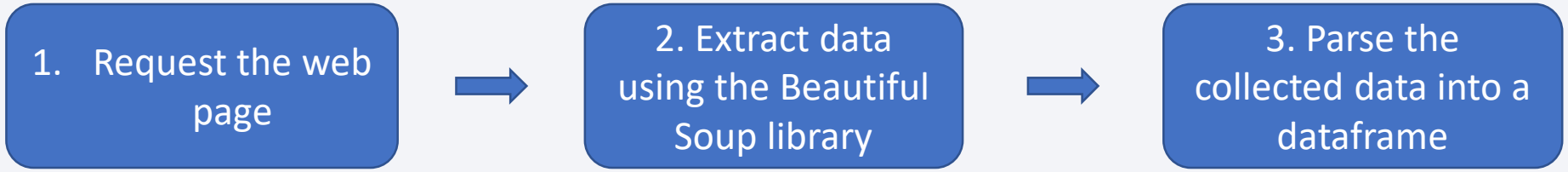
Methodology

Methodology - Executive summary

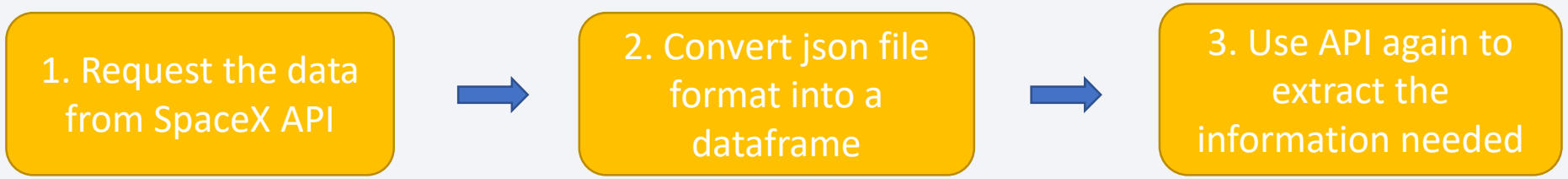
- Data collection methodology:
  - Collect data from Wikipedia. Link
  - Collect data using SpaceX API
- Perform data wrangling
  - Create a new column to show whether landing was successful or not
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Divide data into test and train sets, find the best parameters for the model, fit the data and calculate the accuracy of the model and visualize the result with confusion matrix

Data Collection

- Web scraping from Wikipedia
  - Used to collect information such as: flight nr., launch site, payload, payload mass, orbit, customer, launch outcome, versioon boooster, booster landing, date, and time.



- SpaceX API
  - Used to collect additional information about rocket launches, such as: type of rocket, payload, launch specifications, landing specifications, and landing outcome.



Data Collection – SpaceX API

1. Request the data

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

2. Format the data

```
# Use json_normalize meethod to convert the json result into a dataframe
response=requests.get(static_json_url)
data=pd.json_normalize(response.json())
```

3. Extract the information needed with the customized functions: getBoosterVersion() etc.

4. Create a dataframe

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
# Create a data from launch_dict
dataframe=pd.DataFrame(launch_dict)
```

For more, visit: GitHub URL of the SpaceX API calls notebook

Data Collection - Scraping

1. Request the data

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
obj=requests.get(static_url).text
```

2. Create BeautifulSoup object

```
# Use BeautifulSoup() to create a BeautifulSoup object
soup=BeautifulSoup(obj, 'html5lib')
```

3. Find all tables

```
html_tables=soup.find_all('table')
```

4. Get column names

```
column_names = []
header=first_launch_table.find_all('th')
for i,head in enumerate(header):
    name=extract_column_from_header(head)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

5. Create a dictionary

```
launch dict= dict.fromkeys(column names)
```

6. Add values to the keys

```
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',
   #get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
```

7. Create a dataframe

```
df=pd.DataFrame(launch_dict)
```

For more, visit: GitHub URL of the web scraping notebook

Data Wrangling

- Data processing
  - The main thing in this part of the analysis was to create a new column named 'Class' to show whether the rocket landed successfully or not.

1. Get different landing outcomes

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

2. Create a set of unsuccessful landings

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

3. Encode successful and unsuccesful landings with 1 and 0, respectively

```
landing_class=[]
for outcomes in df['Outcome']:
    if outcomes in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

4. Create a class column

```
df['Class']=landing_class
df[['Class']].head(8)
```

GitHub URL for the data wrangling notebook

EDA with Data Visualization

- The charts used for the visualization of data
  - Scatter plot – used to visualize the trend or relationship between the two variables
  - Bar plot – used to visualize the magnitude/size of the feature/variable
  - Line plot – used to visualize the trend between the two variables. Similar to the scatter plot, only the data points are connected
- GitHub URL for the notebook

EDA with SQL

- The SQL queries were performed according to the following:
  - Display the names of the unique launch sites in the space mission
  - Display 5 records where launch sites begin with the string ‘CCA’
  - Display the total payload mass carried by boosters launched by NASA (CRS)
  - Display average payload mass carried by booster version F9 v1.1
  - List the date where the successful landing outcome in ground pad was achieved
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - List the total number of successful and failure mission outcomes
  - List the names of the booster_versions which have carried the maximum payload mass
  - List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order
- GitHub URL for the EDA with SQL notebook

Build an Interactive Map with Folium

- Used map objects:
  - **Map object** was used to launch the interactive map
  - **Circle object** was used to draw a circle on a specific coordinate area
  - **Marker object** was used to add a launch site name to the circle object on the map
  - **Line object** was used to draw a line between the two coordinate regions on the map
- GitHub URL to the interactive visual analysis with Folium notebook

Build a Dashboard with Plotly Dash

- The graphs on a dashboard
  - **Pie chart** is used to display numerical proportions on a sliced circle. In the created dash application, when choosing ‘All sites’, it will show the proportion of launches made in one specific location out of the total number of launches. When choosing a specific launch site, it will show the proportion of successful and unsuccessful launches for the specific site.
  - **Scatter plot** is used to show the relationship between the two variables. In the created dash application, the payload mass was plotted against the outcome of launch for the different booster versions.
- GitHub URL to Plotly Dash code

Predictive Analysis (Classification)

1. BUILDING THE MODEL:
-Transform the data
-Split the data into train and test sets
-Choose machine learning algorithms to use
-Test different parameter values
-Train the model with the best parameter values with the training data

2. EVALUATING THE MODEL
-Calculate the accuracy score of each model
-Create confusion matrix

3. IMPROVING THE MODEL
-Tune the parameters
-Feature selection

4. CHOOSING THE BEST ALGORITHM
-Check which model gave the best accuracy score

- GitHub URL to predictive analysis notebook

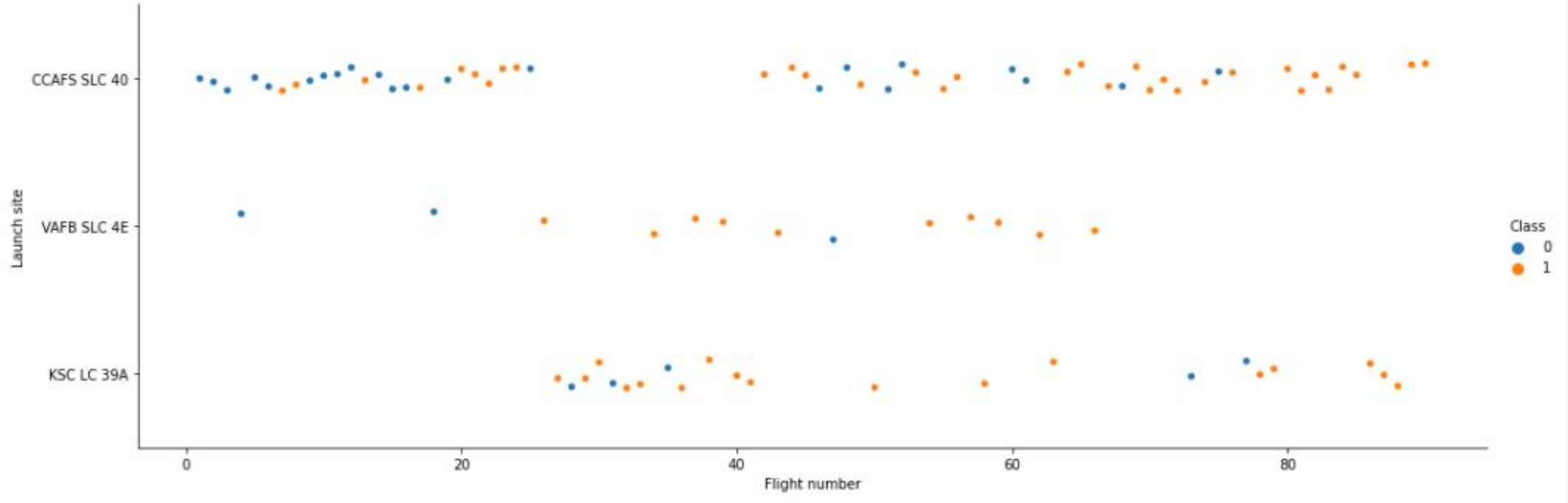
Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

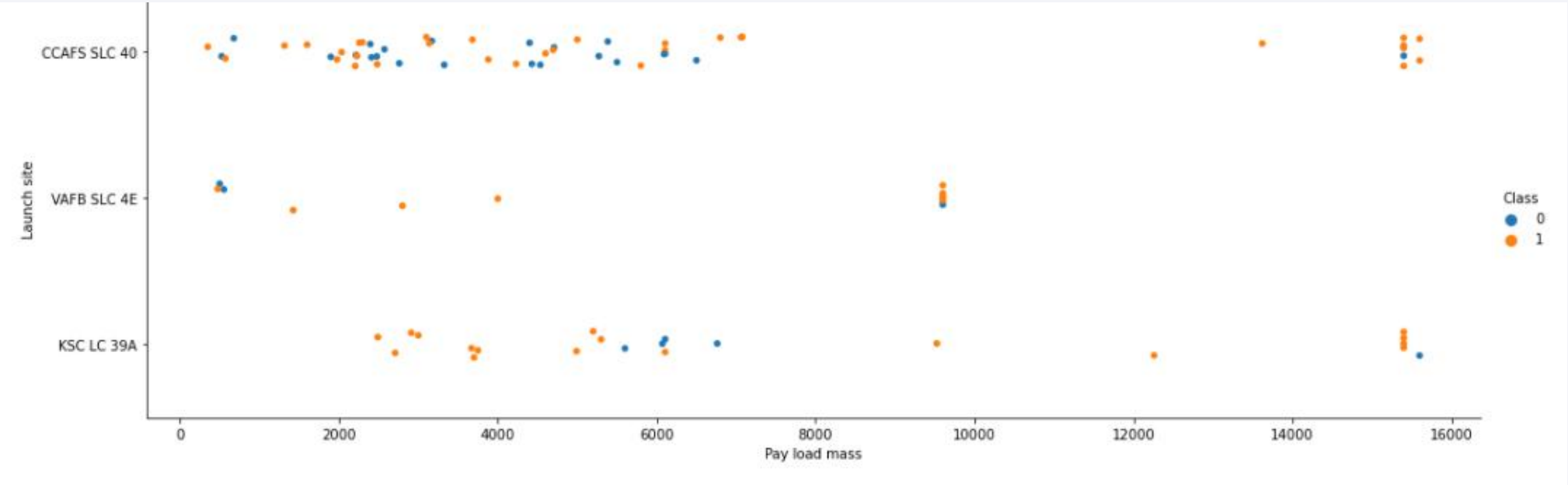
Insights drawn from EDA

Flight Number vs. Launch Site



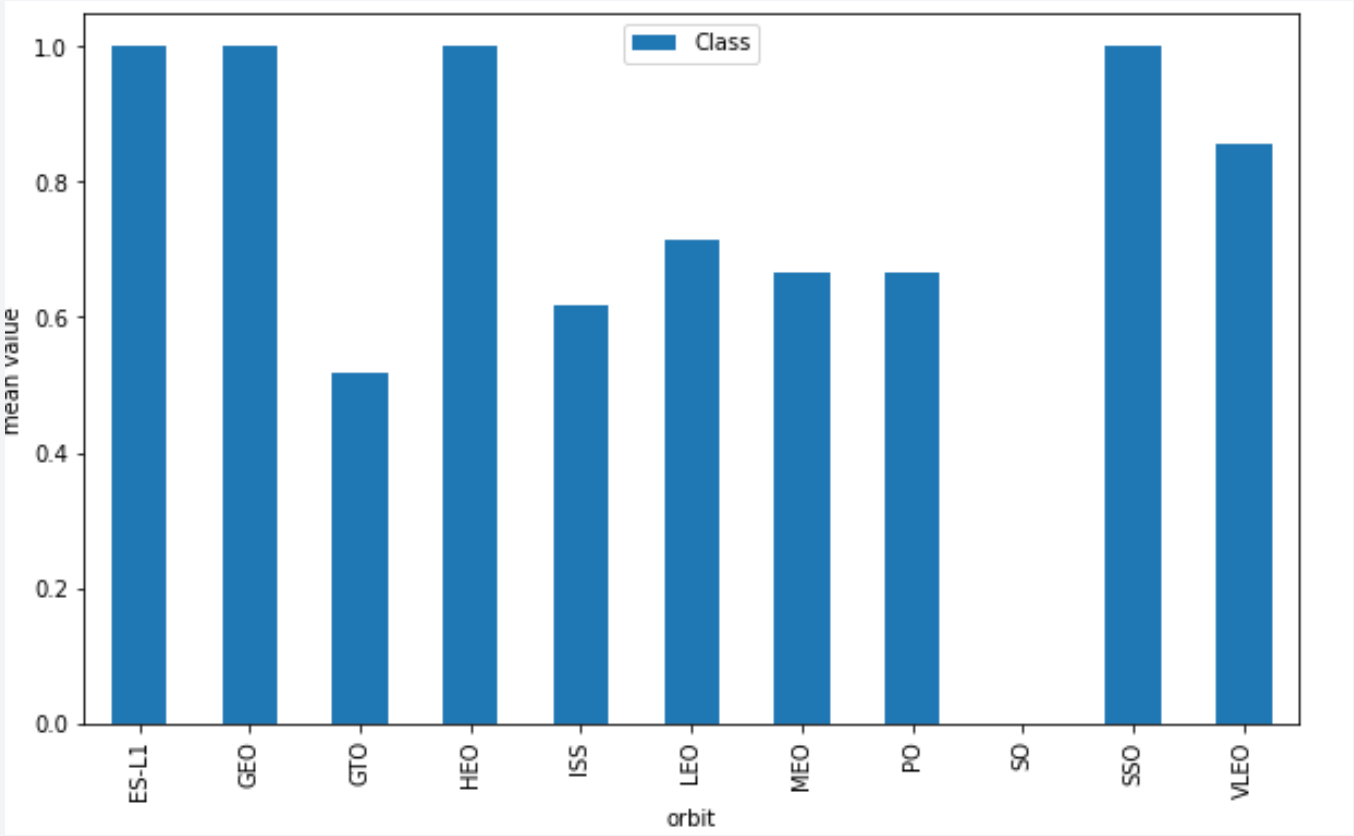
The figure shows that the higher is the flight number, the greater is the change of a successful launching. After the flight number 80 are only successful landings.

Payload vs. Launch Site



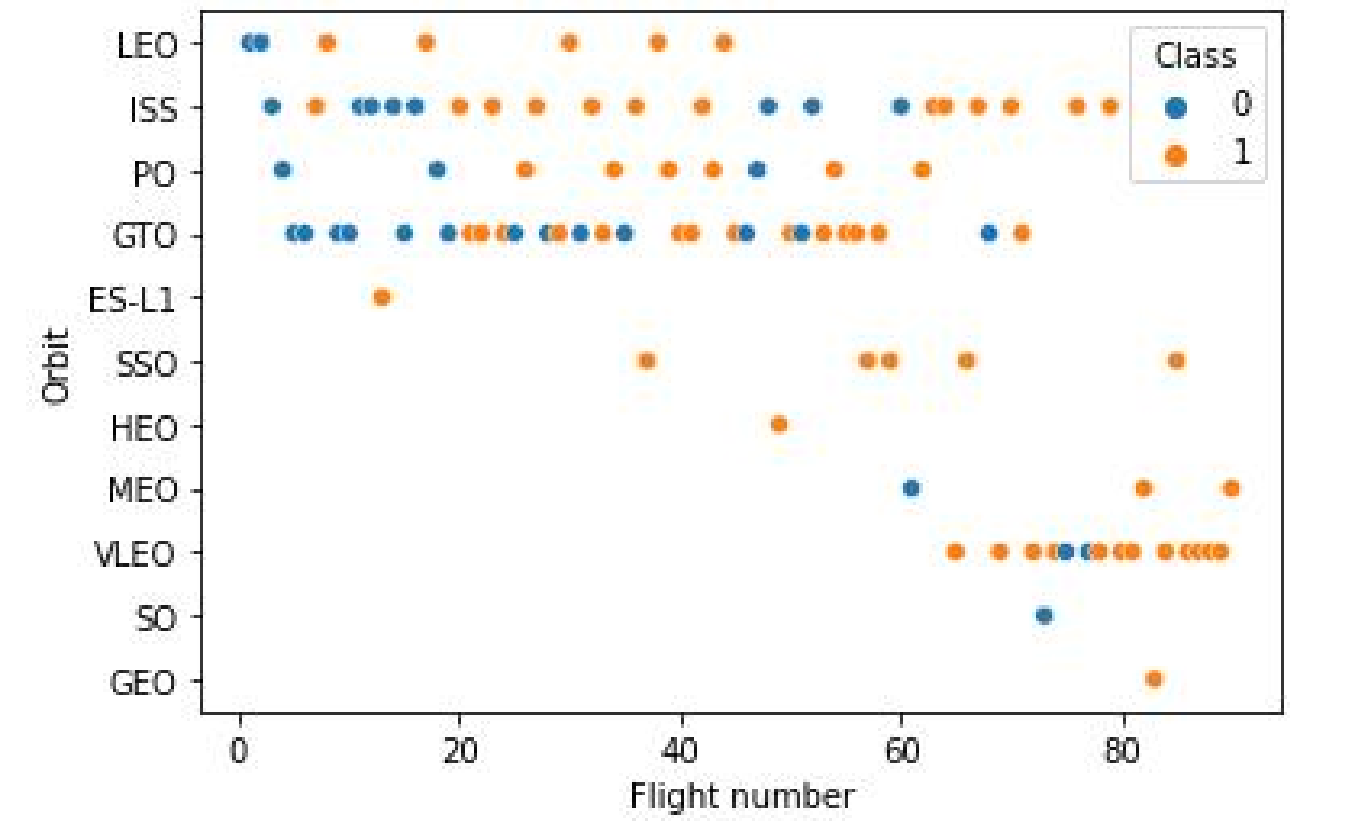
No clear separation between the launch sites and pay load mass is possible to make.

Success Rate vs. Orbit Type



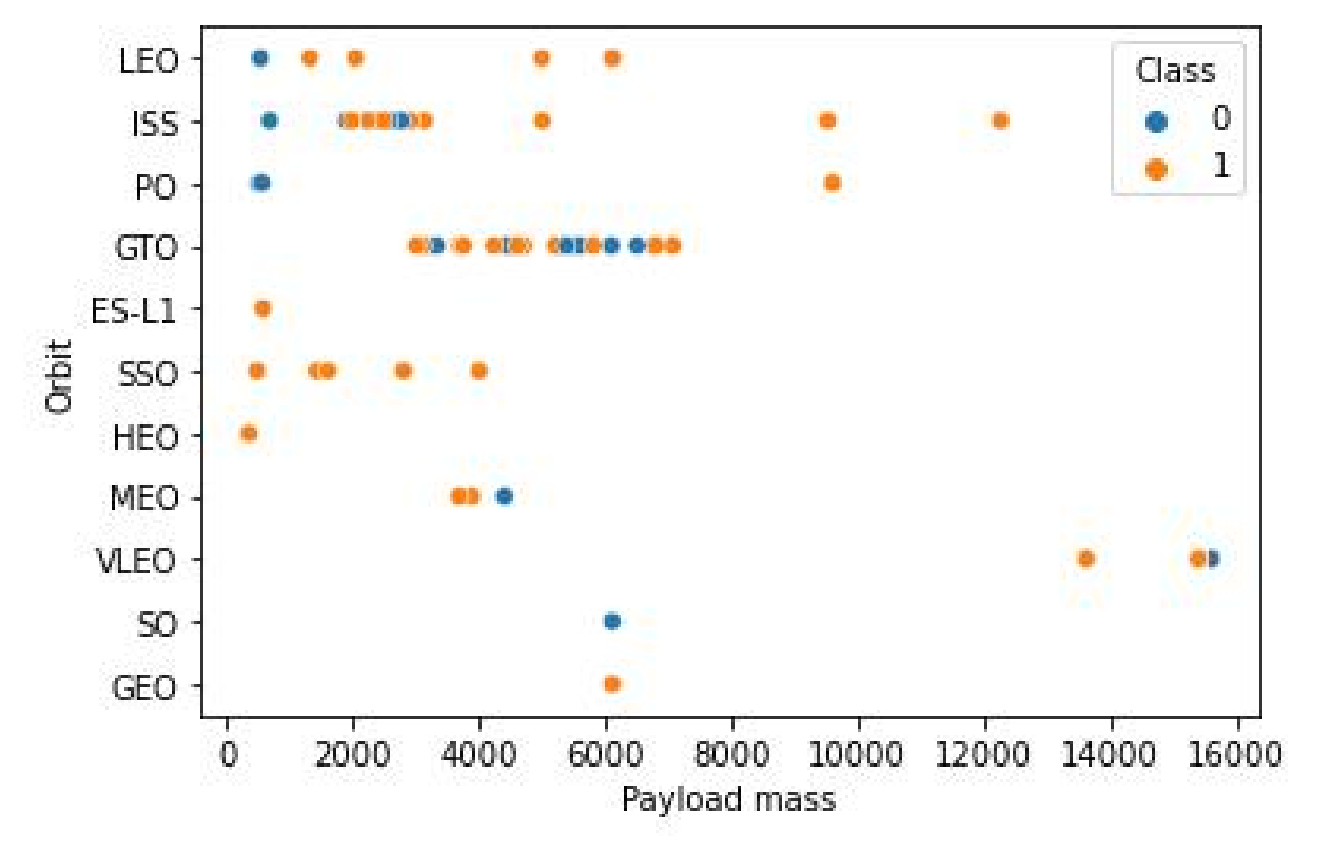
The figure shows that some orbits have the higher average success rate than other orbits. These orbits are ES-L1, GEO, HEO, and SSO.

Flight Number vs. Orbit Type



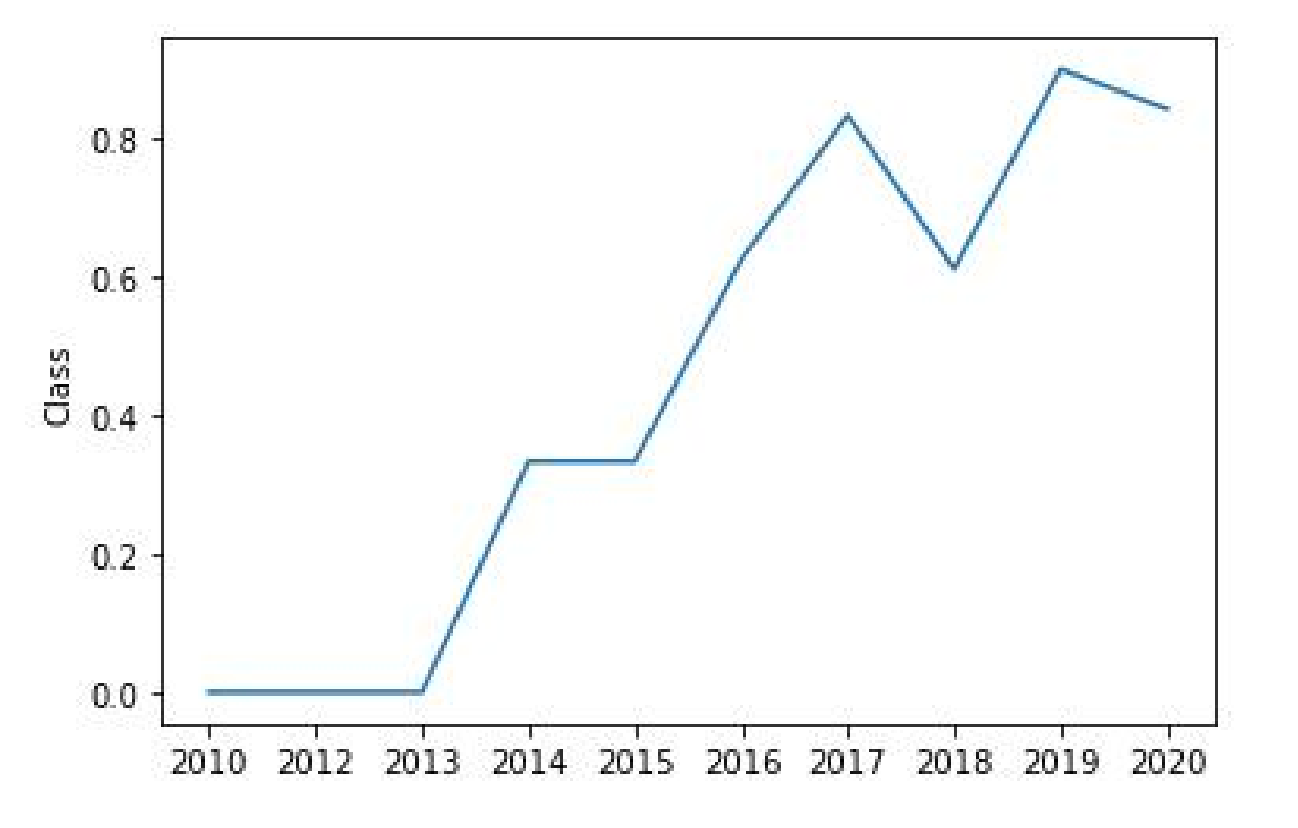
Orbits such as LEO, ISS, SSO, VLEO seem to have an association with the flight number, while GTO has no association with the flight number.

Payload vs. Orbit Type



The figure shows that the heavier payload masses have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

Launch Success Yearly Trend



The figure shows that the average success started to increase in 2013, remains stable in the years 2014-2015, increases again until 2017, decreases in 2018 and then increases again.

All Launch Site Names

- SQL Query:
  - select distinct(launch_site) from SPACEXTBL;
  - The SELECT DISTINCT statement is used to return only distinct values.
  - Output:

launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

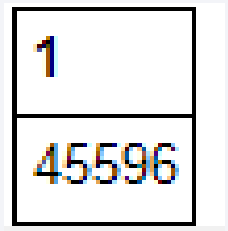
Launch Site Names Begin with 'CCA'

- SQL Query:
  - select * from SPACEXTBL where launch_site like 'CCA%' limit 5;
  - The SELECT * statement is used to retrieve all launch sites starting with ‘CCA’, the output is limited to only 5 rows.
  - Output:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

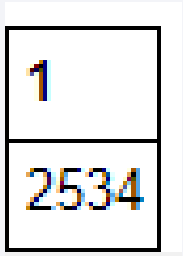
Total Payload Mass

- SQL Query:
  - select sum(payload_mass__kg_) from SPACEXTBL where customer in ('NASA (CRS)');
  - The sum() function returns the total sum of a numeric column, only for a specific column value.
  - Output:



Average Payload Mass by F9 v1.1

- SQL Query:
  - select avg(payload_mass__kg_) from SPACEXTBL where booster_version like 'F9 v1.1%';
  - The avg() function returns the average payload mass for the booster versioon ‘F9 v1.1’
  - Output:



First Successful Ground Landing Date

- SQL Query:
  - select min(date) from SPACEXTBL where landing__outcome in ('Success (ground pad)');
  - The min() function returns the smallest value from the date column where the landing outcome is 'Success (ground pad)'
  - Output:

1
2015-12-22

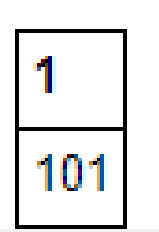
Successful Drone Ship Landing with Payload between 4000 and 6000

- SQL Query:
  - select booster_version from SPACEXTBL where landing__outcome in ('Success (drone ship)') and (payload_mass__kg_ between 4000 and 6000);
  - Select only booster versioon column where landing outcome is ‘Success (drone ship)’ and payload in a specified range.
  - Output:

booster_version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- SQL Query:
  - select count(mission_outcome) from SPACEXTBL where mission_outcome like 'Fail%' or mission_outcome like 'Success%';
  - The count() function returns the number of rows in mission outcome that matches a specified criterion, mission_outcome either Fail or Success.
  - Output:



Boosters Carried Maximum Payload

- SQL Query:
  - select booster_version, (select max(payload_mass__kg_) from SPACEXTBL) from SPACEXTBL;
  - Booster versions with the maximum payload mass are returned.
  - Output:

*not a full list

booster_version	2
F9 v1.0 B0003	15600
F9 v1.0 B0004	15600
F9 v1.0 B0005	15600
F9 v1.0 B0006	15600
F9 v1.0 B0007	15600
F9 v1.1 B1003	15600
F9 v1.1	15600
F9 v1.1	15600
F9 v1.1	15600
F9 v1.1	15600
F9 v1.1	15600
F9 v1.1 B1011	15600
F9 v1.1 B1010	15600
F9 v1.1 B1012	15600
F9 v1.1 B1013	15600
F9 v1.1 B1014	15600

2015 Launch Records

- SQL Query:
  - select landing__outcome, booster_version, launch_site from SPACEXTBL where year(DATE)='2015' and landing__outcome like 'Failure (drone ship)';
  - Return the failed landing_outcomes in drone ship, their booster versions, and launch site names for the year 2015
  - Output:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- SQL Query:
  - select landing__outcome, count(landing__outcome) as count from SPACEXTBL where DATE between '2010-06-06' and '2017-03-20' group by landing__outcome order by count(landing__outcome) desc;
  - Returns the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order
  - Output:

landing__outcome	COUNT
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Uncontrolled (ocean)	2
Failure (parachute)	1
Precluded (drone ship)	1

Section 4

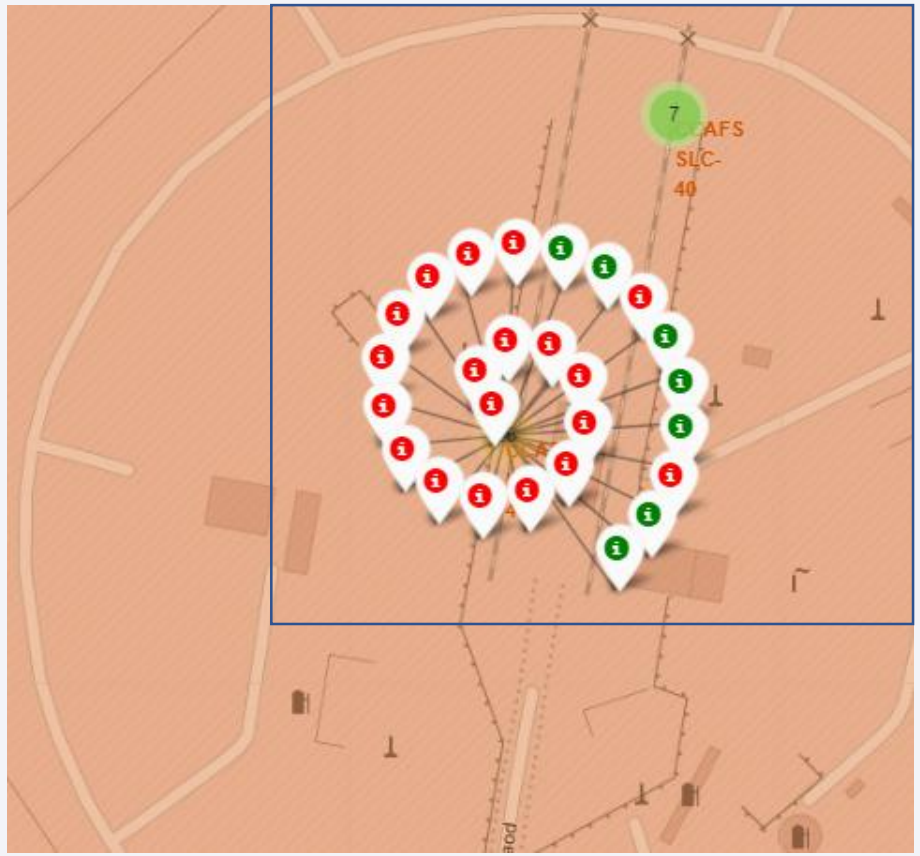
Launch Sites Proximities Analysis

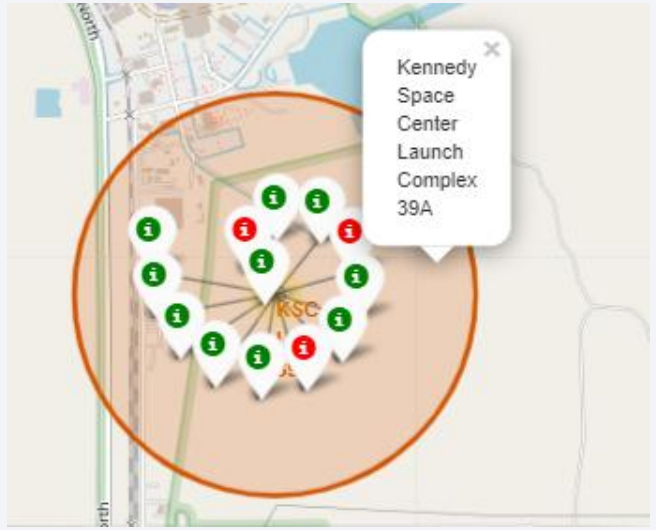
World map with all the launch sites

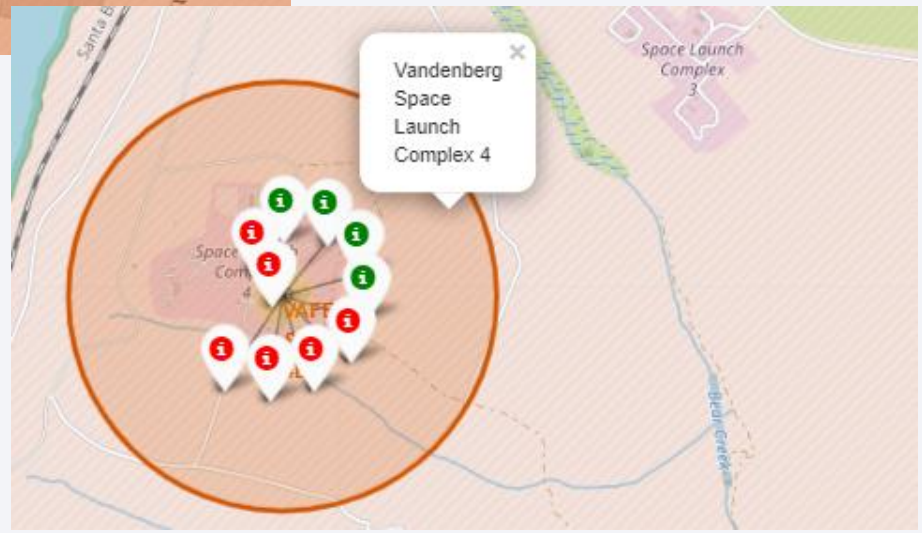
All the SpaceX launch sites are located in North America

Color-labeled launch outcomes for each launch site

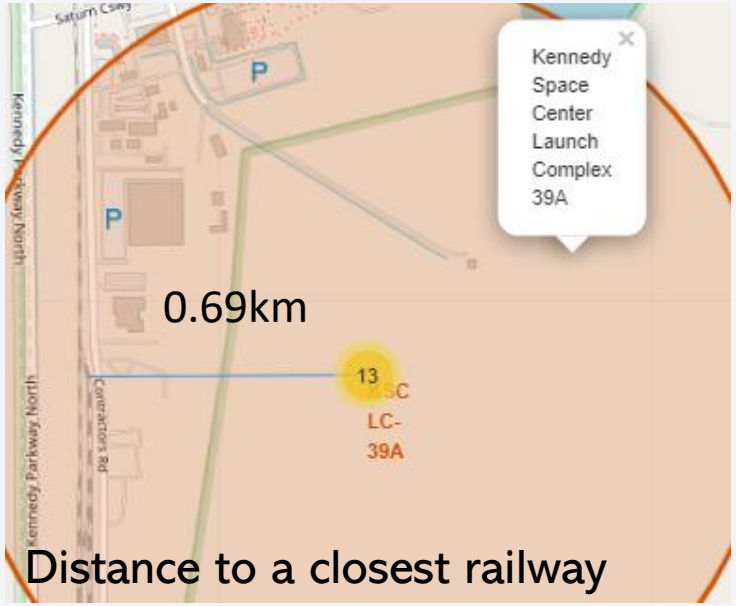
- **Green** – Successful launch
- **Red** – Failed launch



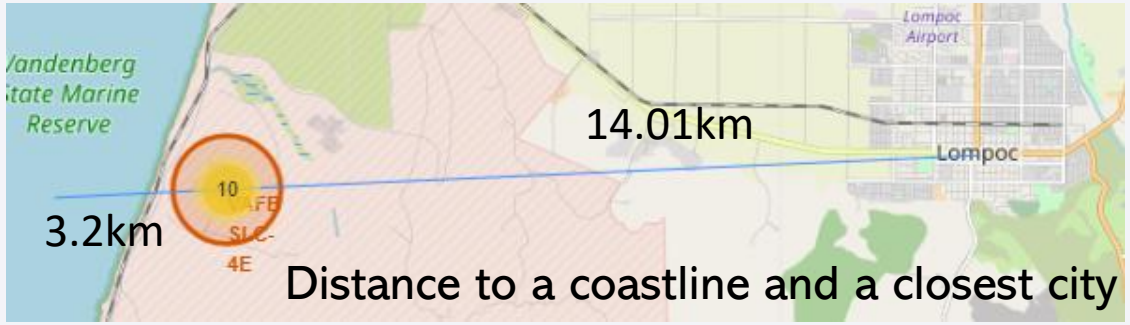




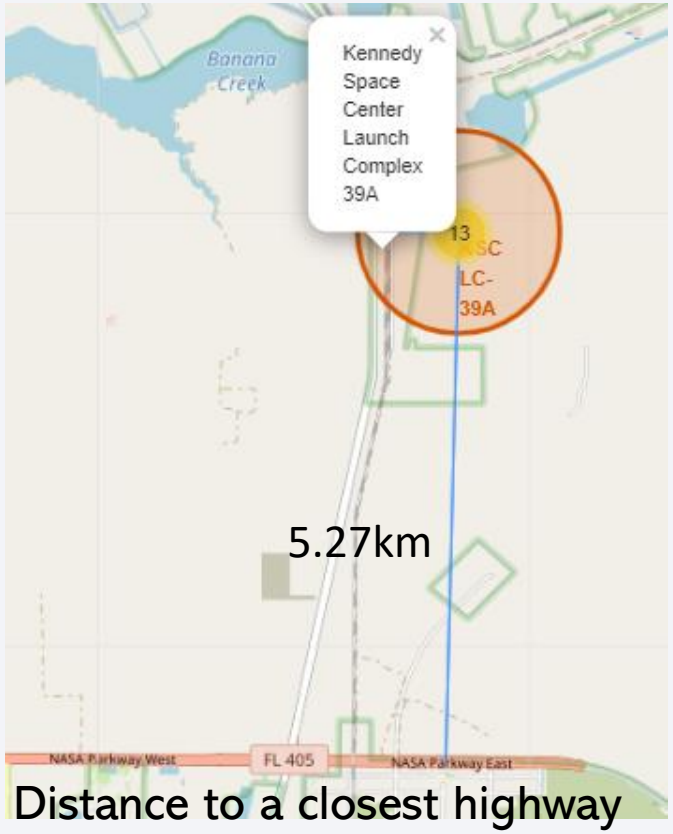
Proximities of launch sites



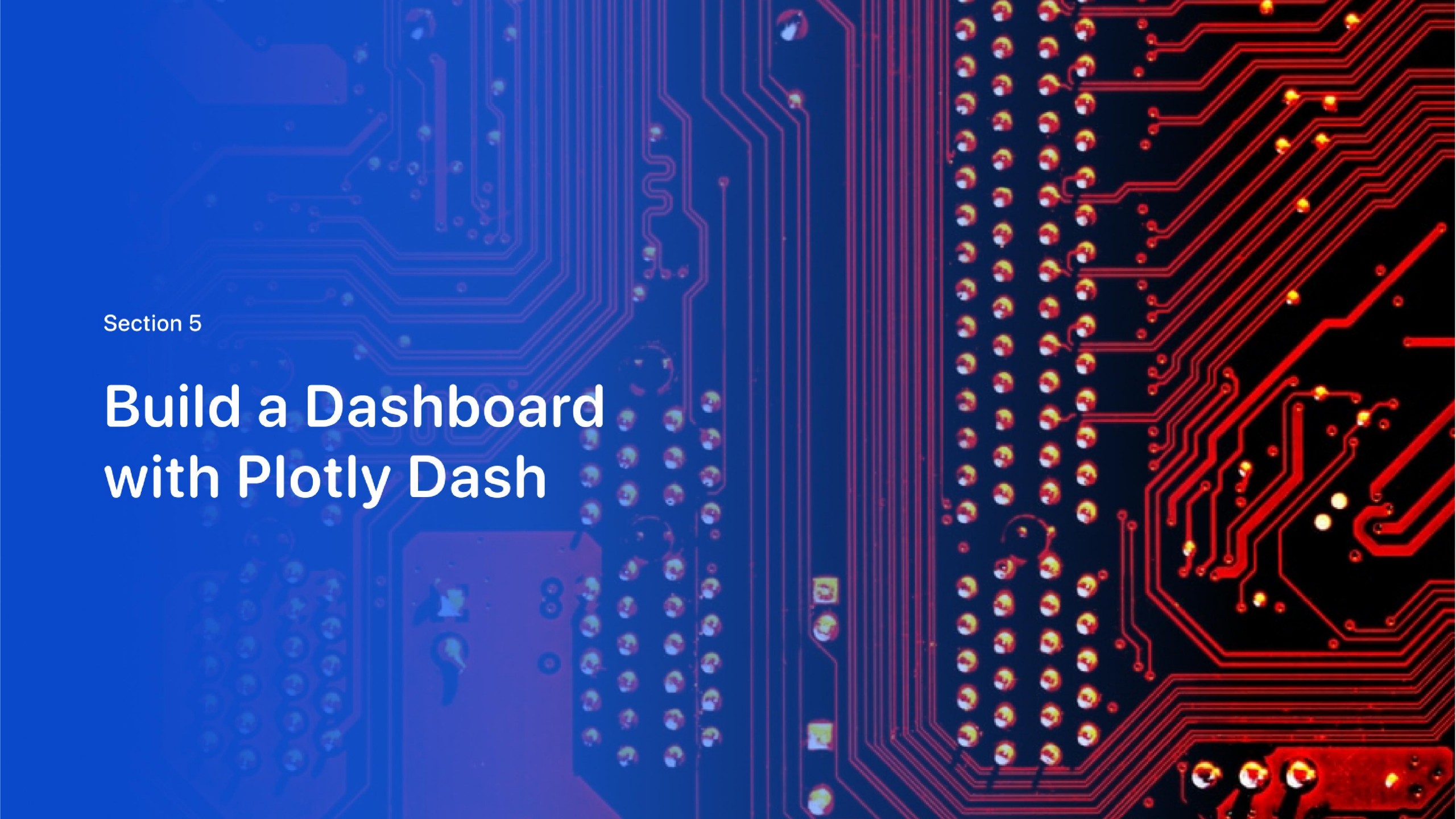
Distance to a closest railway



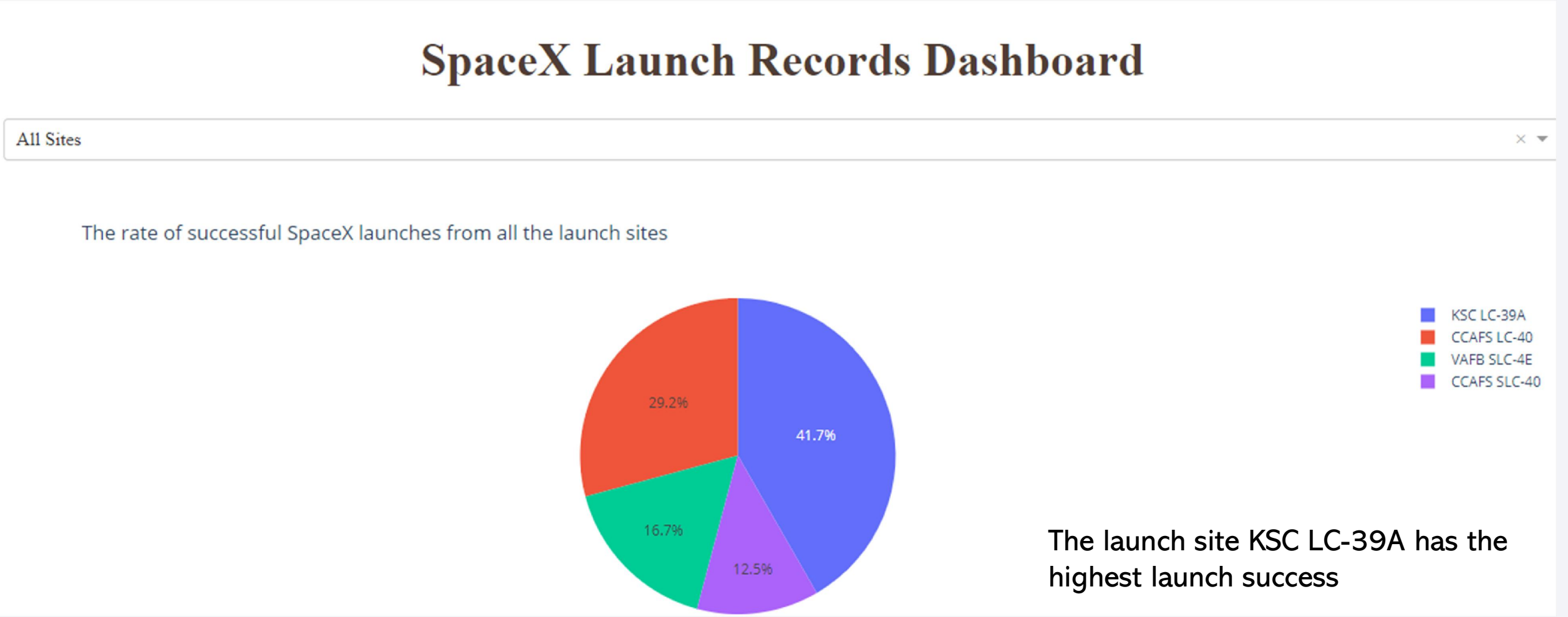
Distance to a coastline and a closest city



Distance to a closest highway

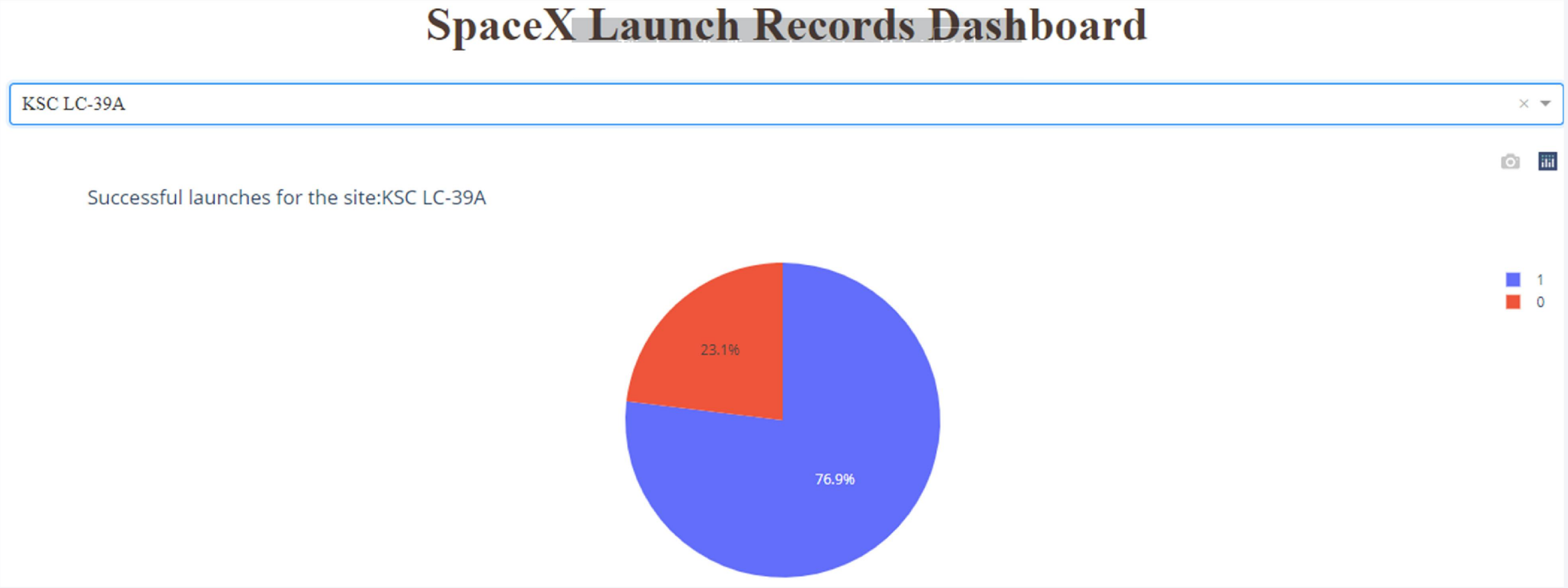

Section 5
Build a Dashboard with Plotly Dash

Launch success for all sites



The launch site KSC LC-39A has the highest launch success

KSC LC-39A launch success ratio



76.9% of the launches were successful and 23.1% of the launches were unsuccessful

Payload vs. Launch Outcome scatter plot for all sites

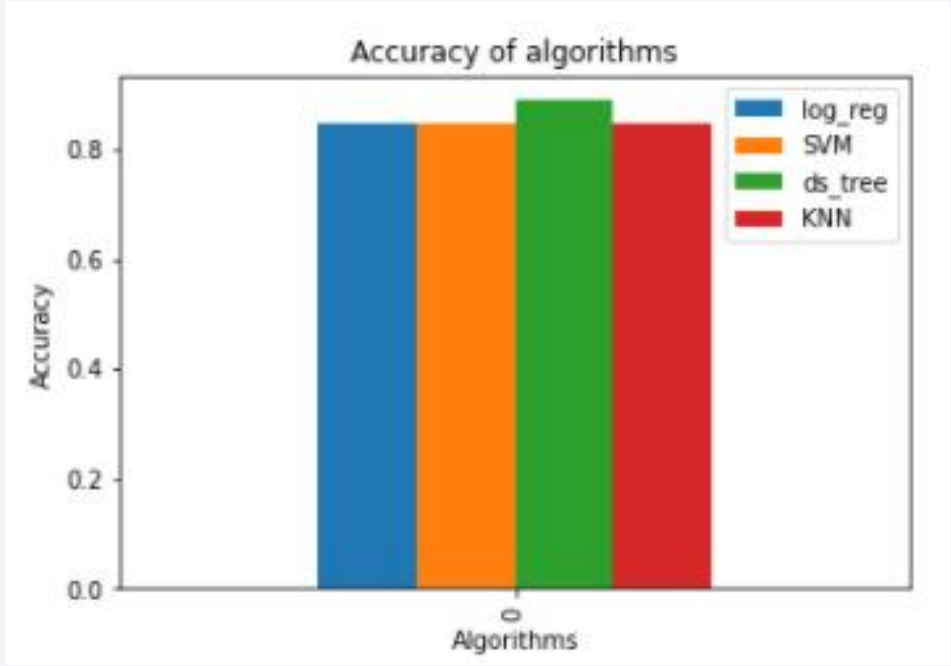
Payload mass between 100 and 6000 contains most of successful launches, the most successful booster version is FT

Section 6

Predictive Analysis (Classification)

Classification Accuracy

- The highest classification accuracy has the decision tree algorithm
  - The accuracy based on the training set was 0.8857
  - The accuracy based on the test set was 0.9444




```
models={'log_reg':logreg_cv.best_score_,'SVM':svm_cv.best_score_,'ds_tree':tree_cv.best_score_,'KNN':knn_cv.best_score_}
best=max(models,key=models.get)
print('The best model is:',best)
print('The accuracy of best model is:',models[best])
```

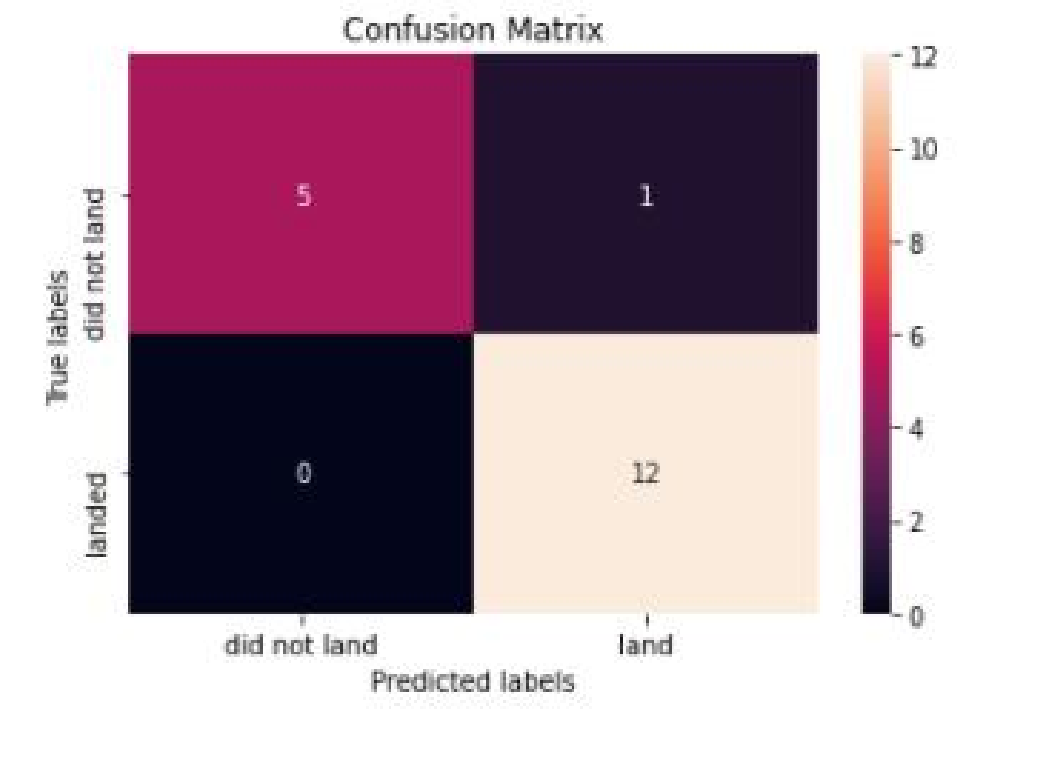
```
The best model is: ds_tree
The accuracy of best model is: 0.8857142857142858
```

```
print('Accuracy of test data:',tree_cv.score(X_test,Y_test))
```

```
Accuracy of test data: 0.9444444444444444
```

Confusion Matrix

- The confusion matrix based on the decision tree model shows 1 false positive value and 17 correcly predicted values



Conclusions

- The orbits ES-L1, GEO, HEO, and SSO have higher success rate
- The pay load mass affects the launch success
- All the observed launch sites are close to the equator and sea, but the KSC LC-39A launch site is the most successful compared to the other lauch sites
- Decision tree model performs the best, showed the highest accuracy in the analysis

Thank you!